Experience with ConSer: A System for Server Control Through Fluid Modeling

Luc Malrait, Sara Bouchenak, Nicolas Marchand

Experience with CONSER: A System for Server Control Through Fluid Modeling

Luc Malrait,Sara Bouchenak, *Member, IEEE,* and Nicolas Marchand

Abstract—Server technology provides a means to support a wide range of online services and applications. However, their ad-hoc configuration poses significant challenges to the performance, availability and economical costs of applications. In this paper, we examine the impact of server configuration on the central tradeoff between service performance and availability. First, we present a server model as a nonlinear continuous-time model using fluid approximations. Second, we develop concurrency control on server systems for an optimal configuration. We primarily provide two control laws for two different QoS objectives. AM-$\mathscr{C}$ is an availability-maximizing server control that achieves the highest service availability given a fixed performance constraint; and PM-$\mathscr{C}$ is a performance-maximizing control law that meets a desired availability target with the highest performance. We then improve the control with two additional multi-level laws. AA-PM-$\mathscr{C}$ is an availability-aware performance maximizing control, and PA-AM-$\mathscr{C}$ is a performance-aware availability maximizing control. In this paper, we present CONSER, a novel system for the control of servers. We evaluate CONSER's fluid model and control techniques on the TPC-C industry-standard benchmark. Our experiments show that the proposed techniques sucessfully guarantee performance and availability constraints.

Index Terms—Server systems, QoS, SLA, Performance, Availability, Modeling, Control.

1 INTRODUCTION

1.1 Context and challenges

A large variety of Internet services exists, ranging from web servers to e-mail servers [1], streaming media services [2], e-commerce servers [3], and database systems [4]. These services are usually based on the classical client-server architecture, where multiple clients concurrently access an online service provided by a server (e.g. reading web pages, sending emails or buying the content of a shopping cart). Such server systems face varying workloads as shown in several studies [5], [6], [7]. For instance, an e-mail server is likely to face a heavier workload in the morning than in the rest of the day, since people usually consult their e-mails when arriving at work. In its extreme form, a heavy workload may induce server thrashing and service unavailability, with underlying economical costs. These costs are estimated at up to US$ 2.0 million/hour for Telecom and Financial companies [8], [9].

A classical technique used to prevent servers from thrashing when the workload increases consists in limiting client concurrency on servers – also known as the multi-programming level (MPL) configuration parameter of servers. This technique is a special case of admission control [10]. Obviously, servers' MPL configuration has a direct impact on server performance, availability and quality-of-service (QoS). Existing approaches to server control either rely on ad-hoc tuning and heuristics without optimality guarantees [11], [12], [13], or apply linear control theory which does unfortunately not capture the intrinsic nonlinear behavior of server systems [14], [15], or follow a queueing theory approach where the system can be accurately modeled but at the expense of a hard model calibration process which makes it unwieldy to use [16], [17], [18]. We believe that modeling server systems is necessary to provide guarantees on the QoS. However, we argue that for the effective deployment of server modeling, the models must accurately capture the *dynamics* and the *nonlinear* behavior of server systems while being *simple* to deploy on existing systems.

- *L. Malrait is with the NeCS Networked Controlled System Research Group, INRIA – Gipsa Lab, Grenoble, France. E-mail: Luc.Malrait@inria.fr*
- *S. Bouchenak is with the SARDES Distributed Systems Research Group, INRIA – Grenoble Universities, France. E-mail: Sara.Bouchenak@inria.fr*
- *N. Marchand is with the NeCS Networked Controlled System Research Group, CNRS – Gipsa Lab, Grenoble, France. E-mail: Nicolas.Marchand@inria.fr*

1.2 Scientific contributions

In this paper, we apply a nonlinear continuous-time control theory based on fluid approximations, in order to model and control the QoS of server systems. The main contribution of the paper is twofold:

- The design and implementation of a nonlinear continuous-time model of server systems that is simple to use since it involves very few external parameters, and which still accurately captures the dynamics of server systems as fluid flows.
- The design and implementation of nonlinear MPL control for server systems. First, two variants of control laws are proposed: AM-$\mathscr{C}$ is an availability-maximizing optimal server control that achieves the highest service availability given a fixed per-

formance constraint, and $PM\text{-}\mathscr{C}$ is a performance-maximizing optimal server control that meets a desired availability target with the highest performance. Furthermore, two additional control laws are proposed for applying performance and availability optimization at multiple levels, the $PA\text{-}AM\text{-}\mathscr{C}$ and $AA\text{-}PM\text{-}\mathscr{C}$ laws.

In this paper, we present the implementation and evaluation of CONSER, a novel system for the control of servers. An evaluation of CONSER was conducted on the TPC-C application, an industry-standard benchmark, running on the PostgreSQL database server. A wide range of application workload conditions was considered. The results of the experiments show that the proposed techniques provide significant benefits on the performance and the availability of the controlled system compared to ad-hoc control solutions.

1.3 Paper roadmap

The remainder of the paper is organized as follows. Section 2 gives an overview of the background. Section 3 presents our contribution in terms of fluid modeling of server systems. Section 4 describes the validation of the proposed model. Sections 5 and 6 respectively present and evaluate the proposed $AM\text{-}\mathscr{C}$ and $PM\text{-}\mathscr{C}$ feedback control laws for servers. Sections 7 and 8 describe and evaluate the $PA\text{-}AM\text{-}\mathscr{C}$ and $AA\text{-}PM\text{-}\mathscr{C}$ control laws for multi-level optimization. Section 9 describes the related work. Finally, Section 10 draws our conclusions.

2 SERVER SYSTEMS

2.1 Definitions

We consider server systems such as database servers and web servers that follow the client-server architecture where servers provide clients with some online service, such as on-line bookstore, or e-banking. Clients and servers are hosted on different computers connected through a communication network. Basically, a client remotely connects to the server, sends it a request, the server processes the request and builds a response that is returned to the client before the connection is closed. Multiple clients may concurrently access the same server.

2.1.1 Server workload

Server workload is characterized, on the one hand, by the number of clients that try to concurrently access a server (i.e. *workload amount*), and on the other hand, by the nature of requests made by clients (i.e. *workload mix*), e.g. read-only requests mix vs. read-write requests mix. Workload amount is denoted as N while workload mix is denoted as M. Furthermore, server workload may vary over time. This corresponds to different client behaviors at different times. For instance, an e-mail service usually faces a higher workload amount in the morning than in the rest of the day.

2.1.2 Server MPL control

Admission control is a classical technique to prevent a server from thrashing [19]. MPL control is a special case of admission control that consists in fixing a limit for the maximum number of clients allowed to concurrently access a server – the Multi-Programming Level (MPL) configuration parameter of a server. Above this limit, incoming client requests are rejected. Thus, a client request arriving at a server either terminates successfully with a response to the client, or is rejected because of the server's MPL limit. Therefore, due to the MPL limit, among the N clients that try to concurrently access a server, only N_e clients actually access the server, with $N_e \leq MPL$. Servers' MPL has a direct impact on the quality-of-service (QoS), performance and availability of servers as discussed below.

2.2 Quality-of-service of server systems

Several criteria may be considered to characterize service performance and availability [13]. In the following, we consider in particular two metrics that reflect performance and availability from the user's perspective [13], namely *latency* and *abandon rate*.

2.2.1 Service performance – Latency

Client request latency is defined as the time needed by the server to process a request. The average client request latency is denoted as L. A low client request latency (or latency, for short) is a desirable behavior which reflects a reactive system. Figure 1 describes the impact of server's MPL value on client request latency, when the workload amount varies [1]. Here, three values of MPL are considered, a low value (1), a medium value (25) and a high value (75). The low MPL is very restrictive regarding client concurrency on the server and thus, keeps the server unloaded and implies a low client request latency. In contrast, with a high MPL, when the server workload amount increases client request latency increases too.

2.2.2 Service availability – Abandon rate

Client request abandon rate is defined as the ratio between requests rejected due to server control and the total number of requests received by a server. It is denoted as α. A low client request abandon rate (or abandon rate, for short) is a desirable behavior that reflects service availability. Figure 2 describes the impact of MPL on client request abandon rate [1]. A low MPL is very restrictive regarding client concurrency on the server, and obviously implies a higher abandon rate compared to a high MPL which accepts more clients.

1. Details on the underlying experimental testbed are given in Section 4.1.

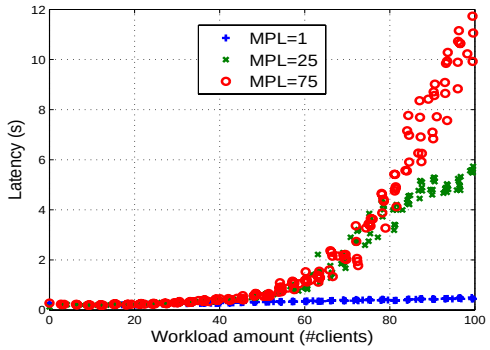


Fig. 1. Impact of MPL on performance

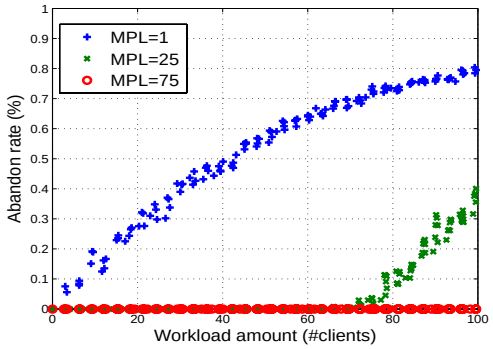


Fig. 2. Impact of MPL on availability

2.2.3 Service Level Agreement

Service Level Agreement (SLA) is a contract negotiated between clients and their service provider [20]. Service performance and service availability are part of the SLA (Service Level Agreement). The SLA specifies the service level objectives (SLOs) such as the maximum latency L_{max} and the maximum abandon rate α_{max} to be guaranteed by the server.

3 ConSer's Fluid Model

We propose a fluid model which renders the dynamics of server systems and captures characteristics that reflect the state of servers in terms of performance and availability. Roughly speaking, fluid approximation consists in looking at all the state variables of the system - that are most integers - as real variables in $\mathbb{R}$. This enables to write the infinitesimal variation of characteristic state variables of the system with respect to time. Those variations can be seen as fluid flows, e.g. client request flows in the present case; and a request queue on the server is similar to a fluid tank [21]. The model is therefore built as a set of differential equations - as for most physical systems in mechanics, physics, electricity, etc. - that describe the time evolution of state variables.

In the present case, we identify three state variables that describe and have an impact on server performance and availability, namely the current number of concurrent client requests in the server N_e, the server throughput T_o and the client request abandon rate α. State variables are usually influenced by themselves and by input variables. The inputs of the proposed model are: the server workload amount N and workload mix M exogenous inputs, and the server MPL tunable parameter that can be used to control the admission to the server. In addition to input and state variables, the model has output variables such as the average latency L to process a client request on the server. In the following, we describe the proposed fluid model through the formulas of its state and output variables.

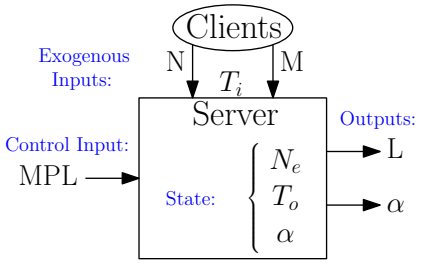


Fig. 3. Model inputs/outputs

3.1 Model state variables

Among the N concurrent clients that try to connect to a server, MPL control authorizes N_e concurrent clients to actually enter the server, with $0 \leq N_e \leq N$ and $0 \leq N_e \leq MPL$. Let $cr(t, t+dt)$ be the number of client connections created on the server between t and $t+dt$, and $cl(t, t+dt)$ be the number of client connections closed on the server between t and $t+dt$. Thus, a balance on N_e between t and $t+dt$ gives

$$N_e(t+dt) = N_e(t) + cr(t, t+dt) - cl(t, t+dt) \quad (1)$$

Let T_i be the incoming throughput of the server, measured as the number of client connection demands per second. It comes that the number of connections created between t and $t+dt$ is

$$cr(t, t+dt) = (1 - \alpha(t)) \cdot T_i(t) \cdot dt \quad (2)$$

where α is the abandon rate of the server.

Similarly, let T_o be the outgoing throughput of the server, measured as the number of client requests a server is able to handle per second. Thus, the number of connections closed between t and $t+dt$ is

$$cl(t, t+dt) = T_o(t) \cdot dt \quad (3)$$

Deriving from (1), (2) and (3), we have $\dot{N}_e$, the derivative of N_e

$$\dot{N}_e(t) = (1 - \alpha(t)) \cdot T_i(t) - T_o(t) \quad (4)$$

Moreover, we assume that the system reaches a steady state in a reasonably short period of time Δ; this is particularly reflected in state variables outgoing throughput T_o and abandon rate α. During this short period of time, the workload is relatively stable, which is consistent with

studies such as [7]. Thus, the dynamics of T_o and α can be approximated by first order systems through their derivatives as follows

$$\dot{T}_o(t) = -\frac{1}{\Delta}\left(T_o(t) - \bar{T}_o\right)$$
$$\dot{\alpha}(t) = -\frac{1}{\Delta}\left(\alpha(t) - \bar{\alpha}\right)$$

where $\bar{T}_o$ and $\bar{\alpha}$ are the steady state values of respectively the outgoing throughput and the abandon rate of the server. The next step naturally consists in finding the expression of $\bar{T}_o$ and $\bar{\alpha}$. A balance on the number of served client requests (or outgoing requests) N_o gives

$$N_o(t+dt) = No(t) + sr(t, t+dt)$$

where $sr(t, t+dt)$ is the number of served request between t and $t+dt$. Since there are N_e concurrent clients on the server and the average client request latency is L, the number of served requests during dt will be $sr(t, t+dt) = \frac{dt}{L}N_e$. Thus, we get $\dot{N}_o = \frac{N_e}{L}$, that is $\bar{T}_o = \frac{N_e}{L}$ which is an expression of Little's law [22].

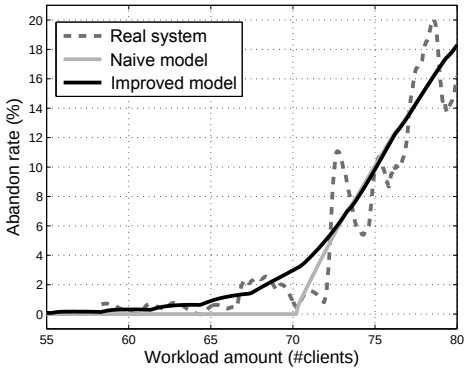


Fig. 4. Accuracy of modeled abandon rate

By definition, $\bar{\alpha}$ is equal to zero if N_e is smaller than MPL, and $\bar{\alpha}$ is equal to $1-\frac{T_o}{T_i}$ if $N_e = MPL$ (see Figure 4, naive model). However, the stochastic nature of the client request arrival may lead to situations where the measured average N_e is smaller than MPL but where punctually, the number of clients that try to access the server is actually higher than MPL, and thus, some clients are rejected. This is illustrated in Figure 4 which compares the actual measured abandon rate with the naive estimation of the abandon rate, showing a mismatch between the two [2] . In order to take this behavior into account, we choose to write $\bar{\alpha} = \frac{N_e}{MPL} \cdot \left(1 - \frac{T_o}{T_i}\right)$. This renders that the probability to reject a client connection is higher when the average N_e is close to MPL. Figure 4 shows that this improved method provides a more accurate estimation of the abandon rate. Finally, it follows that

$$\dot{T}_o(t) = -\frac{1}{\Delta}\left(T_o(t) - \frac{N_e(t)}{L(t)}\right) \quad (5)$$
$$\dot{\alpha}(t) = -\frac{1}{\Delta}\left(\alpha(t) - \frac{N_e(t)}{MPL(t)} \cdot \left(1 - \frac{T_o(t)}{T_i(t)}\right)\right) \quad (6)$$

3.2 Model output variables

Now that we have defined the model state variables, the last step consists in expressing the model output variable latency L. Latency obviously depends on the global load of the server, i.e. the workload mix M and the number of concurrent clients on the server N_e. Figure 5 describes the evolution of latency L as a function of N_e, for a given workload mix [2]. One can see that a second degree polynomial in N_e is a good approximation of the latency L. Thus:

$$L(N_e, M, t) = a(M,t)N_e^2 + b(M,t)N_e + c(M,t) \quad (7)$$

The parameter c is positive as it represents the zero-load latency. a and b are also positive since they model the processing time of requests.

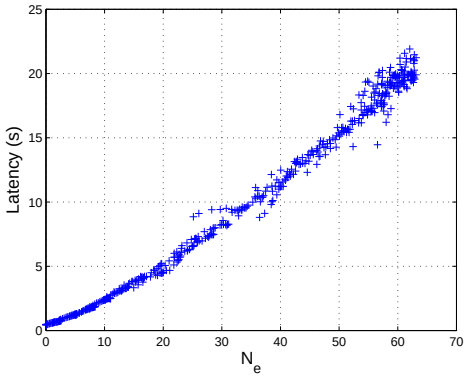


Fig. 5. Latency as a function of N_e

In summary, the proposed fluid model is given by equations (4) to (7) that reflect the dynamics of the state and outputs of server systems in terms of performance and availability. Section 5 then describes the proposed control techniques that build upon the fluid model in order to guarantee service performance and availability level objectives.

4 Model Validation

This section first describes the environment that underlies our experiments, before presenting the results of the evaluation of the proposed fluid model.

4.1 Experimental setup

The evaluation of the proposed fluid model has been conducted using the TPC-C benchmark [23]. TPC-C is an industry standard benchmark from the Transaction Processing Council that models a realistic database server

2. Details on the underlying experimental testbed are given in Section 4.1.

application as a warehouse system where clients request transactions on warehouses stored on a database server. TPC-C comes with a client emulator which emulates a set of concurrent clients that remotely send requests to the database server. The TPC-C client emulator allows to specify the number of concurrent clients to launch (i.e. the workload amount N). It also specifies the client think time, that is the interarrival time between two consecutive client requests. We extended the client emulator in order to be able, on the one hand, to vary the workload amount N over time, and on the other hand, to vary the workload mix M over time. For the latter extension, we considered two mixes of workload, one consisting of read-only requests, and another consisting of read-write requests.

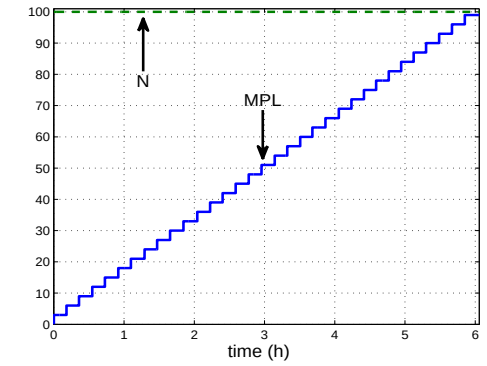
Our experiments have been conducted on a set of two computers connected via a 100 Mb/s Ethernet LAN, one computer dedicated to the database server and another to the client emulator. The database server is PostgreSQL 8.2.6 [4]. The proposed model was implemented using an online monitoring of the system which allows to maintain the state of the model. Well-known Kalman filtering techniques were therefore applied [24]. Both client and server machines run Linux Fedora 7. The server machine is a 3 GHz processor with 2GB RAM, while the clients' computer is a 2 GHz processor with 512MB RAM.

4.2 Real system vs. modeled system

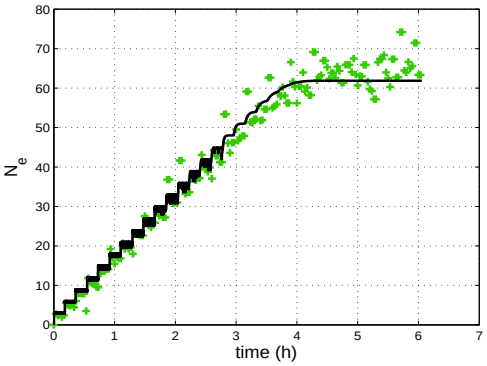
We perform measurements to validate the accuracy of the proposed fluid model and its ability to render the dynamics of the system. In particular, we evaluate the ability of the model to reflect the variation of the state of the system when input variables such as the server MPL and the workload amount N vary. The variation of the state of the system is described by the state variables N_e for the number of concurrent clients admitted in the server, T_o for the outgoing throughput of the server, and α for the client request abandon rate. Thus, for the same set of input variables, the state reified by the model is compared with the actual state of the real system.

Figure 6 describes the case of an open loop system where the workload amount N trying to access the database server is fixed (to 100 clients) and where the MPL value of the server varies (see Figure 6(a)). Figures 6(b), 6(c) and 6(d) show the evolution over time of respectively the number N_e of concurrent clients admitted in the server, the outgoing throughput T_o and the abandon rate α, for both the real system (+) and the model (solid line). Results show that the model accurately reflects the behavior of the real system. For instance, we can observe a thrashing phenomenon of the server when T_o decreases whereas N_e increases. And the model is able to render that behavior, which would not be possible without an overlinear term with respect to N_e in Equation (7).

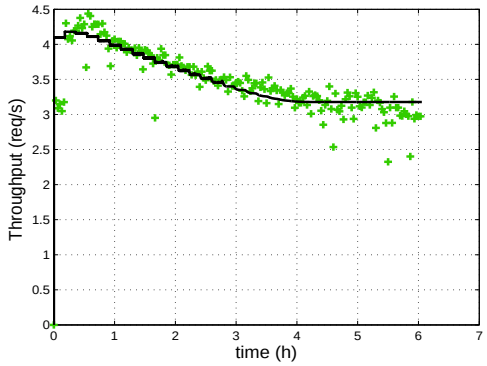
Figure 7 illustrates the case of a dynamic open loop system where both the workload amount N and the



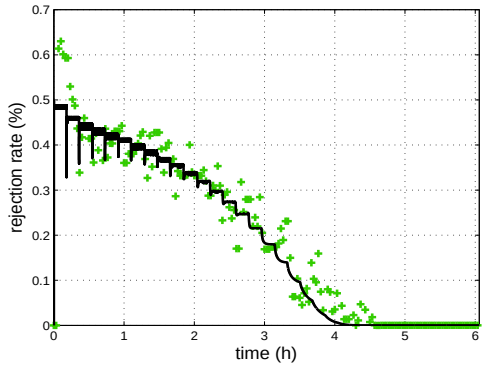
(a) Varying MPL with a fixed workload amount



(b) Admitted concurrent clients



(c) Throughput



(d) Abandon rate

Fig. 6. System behavior with a varying MPL and a fixed workload amount – Real system (+) vs. modeled system (solid line)

server MPL vary over time (see Figure 7(a)). Figures 7(b), 7(c) and 7(d) present the evolution over time of respectively the number N_e of concurrent clients admitted in the server, the outgoing throughput T_o and the abandon rate α, for both the real system (+) and the modeled system (solid line). Results show that the model is able to render the behavior of the real system.

5 AM-$\mathscr{C}$ AND PM-$\mathscr{C}$ CONTROL LAWS

In the following, we study the tradeoff between the performance and the availability of server systems, and derive the optimal MPL control of server systems based on the proposed fluid model, that is the optimal number of concurrent clients admitted to the server with respect to this tradeoff. In particular, we provide two variants of control laws, namely AM-$\mathscr{C}$ and PM-$\mathscr{C}$. AM-$\mathscr{C}$ is an availability-maximizing optimal server MPL control that achieves the highest service availability given a fixed performance constraint. Symmetrically, PM-$\mathscr{C}$ is a performance-maximizing optimal server MPL control that meets a desired availability target with the highest performance. In the present case, service availability is measured as the client request acceptation rate (i.e. 1 - α), and service performance is measured as the average client request latency (i.e. L).

5.1 AM-$\mathscr{C}$ availability-maximizing control

AM-$\mathscr{C}$ aims at guaranteeing a tradeoff between server performance and availability with the following properties:

(P1) the average client request latency does not exceed a maximum latency $L_{\max}$, and
(P2) the abandon rate α is made as small as possible.

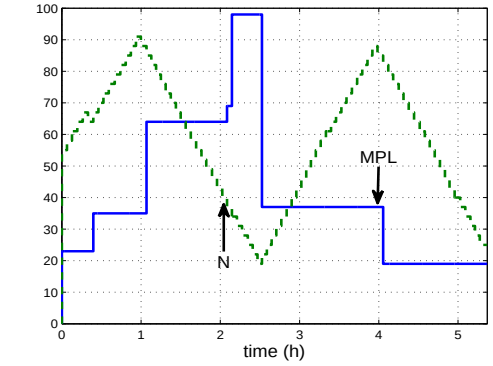
To that end, a feedback control law is proposed to automatically adjust the MPL server control parameter in order to satisfy this tradeoff. The basic idea behind this law is to admit clients in such a way that the average client request latency L is close (equal) to $L_{\max}$. By construction, this maximizes the number of admitted clients N_e, which induces a minimized abandon rate α.

A first approach could consist in solving Eq. (7) in such a way that $L = L_{\max}$. Although accurately reflecting the system, such an approach is unwieldy since it requires the knowledge of accurate values of parameter a, b and c in equation 7, through an online identification of these parameters since the workload may change over time.

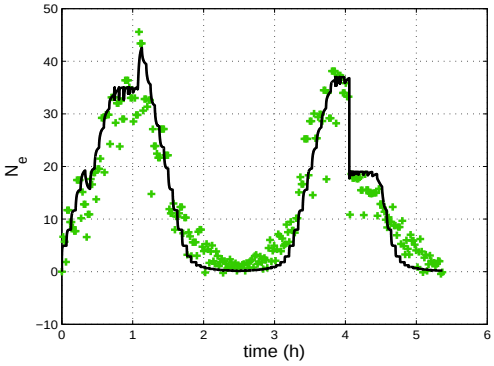
We propose another approach which avoids this online identification of model's parameters. It is obtained via a simple input-output linearization technique in which the considered output is latency L [25]. Roughly speaking, the approach aims at determining how to control the MPL value in such a way that

$$\dot{L} = -\gamma_L (L - L_{\max}) \quad (8)$$

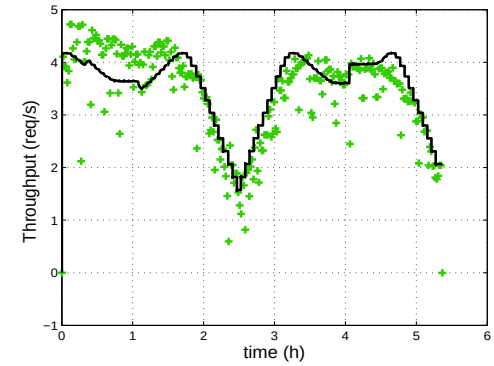
As soon as $\gamma_L > 0$, this will ensure the convergence of L to its maximum $L_{\max}$. From Eq. (7), we have



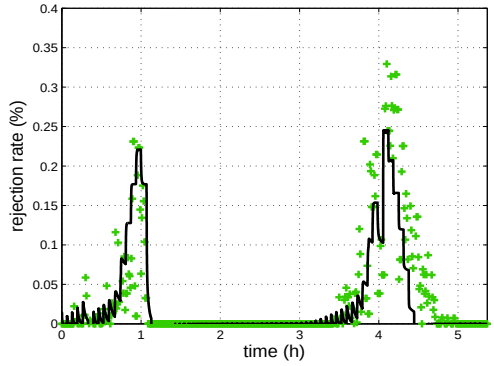
(a) Varying MPL and workload amount



(b) Admitted concurrent clients



(c) Throughput



(d) Abandon rate

Fig. 7. System behavior with varying MPL and workload amount – Real system (+) vs. modeled system (solid line)

$\dot{L} = (2aN_e + b)\,\dot{N}_e$. And since T_o and α reach a steady state in a reasonably short period of time, $T_o(t) = \bar{T}_o$ and $\alpha(t) = \bar{\alpha}$ Therefore, with Eq. (4) we have

$$\dot{L} = (2aN_e + b)\left(1 - \frac{N_e}{MPL}\right)(T_i - \bar{T}_o) \tag{9}$$

As a result from Eq. (8) and (9), MPL should be controlled as follows

$$MPL = \frac{N_e}{1 + \frac{\gamma_L}{(2aN_e+b)(T_i-\bar{T}_o)}(L - L_{\max})}$$

To free ourselves from a and b, we choose to use $\gamma'_L = \frac{\gamma_L}{(2aN_e+b)(T_i-\bar{T}_o)}$, which produces

$$MPL = \frac{N_e}{1 + \gamma'_L(L - L_{\max})} \tag{10}$$

where $\gamma'_L > 0$ is a tuning parameter. It follows that with Eq. (8) and control described in (10), the dynamic evolution of L is given by:

$$\dot{L} = -\left(\gamma'_L\,(2aN_e + b)\,(T_i - \bar{T}_o)\right)(L - L_{\max})$$

Here again, L will converge to $L_{\max}$.

In summary, it is interesting to notice that the feedback control law given in (10) will reflect one of the following situations. If the current latency L is higher than L_{max}, property (P1) is not guaranteed and the control law will produce an MPL as a decreased value of the current number of admitted concurrent clients N_e (since $(1 + \gamma'_L(L - L_{\max})) > 1$), which aims at meeting (P1). Symmetrically, if L is lower than L_{max}, property (P1) holds but property (P2) may not hold, and the control law will produce an MPL as an increased value of N_e (since $(1 + \gamma'_L(L - L_{\max})) < 1$), which aims at meeting (P2). Finally, if L is equal to L_{max}, both properties (P1) and (P2) hold.

Moreover, we observe that with the highest value of γ_L the MPL reaches its highest value while keeping the latency near its authorized limit. Thus, with the AM-$\mathscr{C}$ control law, the highest value of γ_L to be used is $1/L_{max}$.

5.2 PM-$\mathscr{C}$ performance-maximizing control

Similarly, PM-$\mathscr{C}$ aims at guaranteeing the following tradeoff between server performance and availability where:

(P3) the client request abandon rate does not exceed a given maximum abandon rate $\alpha_{\max}$,

(P4) with the lowest average client request latency.

In this context, (P4) will be ensured given (P3) iff the MPL converges to the smallest value that guarantees $\alpha \leq \alpha_{max}$. Once again, we use an input-output linearization approach, taking α as the output, to solve the problem

$$\dot{\alpha} = -\gamma_\alpha(\alpha - \alpha_{\max}) \tag{11}$$

with $\gamma_\alpha > 0$. Furthermore, since the workload remains relatively stable during a short period of time, as stated previously, $\dot{N}_e = 0$. Then, from Eq. (4) and (6), we get

$$\alpha = 1 - \frac{T_o}{T_i}$$

$$\dot{\alpha}(t) = -\frac{1}{\Delta}\alpha(t)(1 - \frac{N_e(t)}{MPL(t)}) \tag{12}$$

Thus, from Eq. (11) and (12) and with the following control applied to MPL, α will converge to α_{max}

$$MPL = \frac{\alpha N_e}{\alpha - \gamma'_\alpha(\alpha - \alpha_{\max})} \tag{13}$$

where $\gamma'_\alpha = \gamma_\alpha \Delta$.

We observe that with the highest value of γ_α the MPL reaches its highest value while keeping the abandon rate under its authorized limit. Thus, with the PM-$\mathscr{C}$ control law, the highest value of γ_α to be used is $1/(1 - \alpha_{max})$.

6 AM-$\mathscr{C}$ AND PM-$\mathscr{C}$ EVALUATION

This section presents the results of the evaluation of the implemented feedback controllers presented in Section 5 when applied to the PostgreSQL database server that hosts the TPC-C database. The results of the experiments conducted with the AM-$\mathscr{C}$ availability-maximizing controller are first presented in Section 6.2, and the results of the PM-$\mathscr{C}$ performance-maximizing controller are then described in Section 6.3.

6.1 Experimental environment

We used the same experimental environment as the one described in Section 4.1. The proposed controllers were deployed as follows. A proxy-based approach was followed to implement the AM-$\mathscr{C}$ and PM-$\mathscr{C}$ controllers where a proxy stands in front of the database server to implement online feedback server control. Moreover, the CONSER-based controlled system is compared with two base systems applying ad-hoc MPL control, that is ad-hoc control 1 with a static MPL set to 25 and ad-hoc control 2 with a static MPL at 40.

6.2 AM-$\mathscr{C}$ evaluation

In this section, we evaluate the proposed AM-$\mathscr{C}$ availability-maximizing feedback controller presented in Section 5.1. Here, we consider a performance constraint limiting the maximum average client request latency to 8 s. The role of AM-$\mathscr{C}$ is thus to guarantee that performance constraint while maximizing service availability, through online feedback control of the server MPL. We consider two scenari to evaluate this controller, each one illustrating a variation of one of the two exogeneous input variables of the system, i.e. the first scenario considers a changing workload mix, and the second scenario handles a varying workload amount N.

6.2.1 Workload mix variation

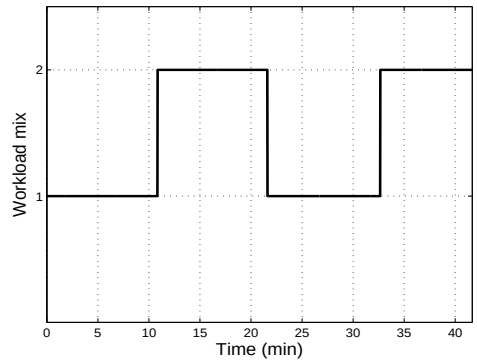
Figure 8 describes the first scenario where the workload mix varies from $M1$ to $M2$ twice (c.f. Figure 8(a)), while the workload amount N is of 80 clients. The workload mix $M1$ consists of read-only requests while the workload mix $M2$ generates read-write requests. The two mixes differ in their average request latency as follows. With 10 concurrent clients in the server, the average client request latency is $0.23s$ with mix $M1$ and $0.55s$ with mix $M2$. Figures 8(d), 8(b) and 8(c) present the variation over time of respectively the server MPL, the average client request latency and the client request abandon rate. The figures compare the two base systems using ad-hoc control 1 and ad-hoc control 2 with the CONSER-based controlled system. Notice that the sudden change of MPL after the 10th, 20th and 30th minutes corresponds to workload mix changes; this has also an impact on latency and abandon rate.

Results demonstrate that the AM-$\mathscr{C}$ controller is able to dynamically adjust MPL in order to guarantee the latency performance constraint while keeping the service availability to its maximum, with an abandon rate minimized to 0% with $M1$ and to 10% in average with $M2$. Whereas none of the two base systems with ad-hoc control is able to guarantee the SLOs when the workload varies. Here, compared to CONSER, a latency overhead of up to 25% is induced by ad-hoc control 2 and an abandon rate overhead of up to 28% results from ad-hoc control 1.

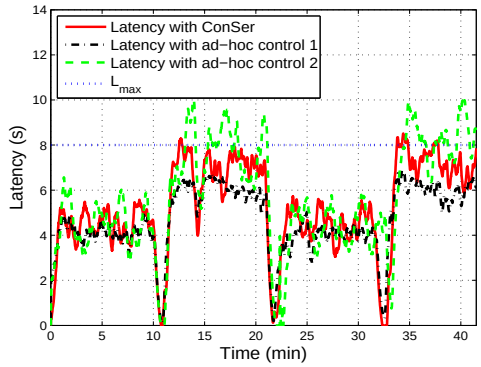
6.2.2 Workload amount variation

Figure 9 presents another dynamics of the system, that is the variation of the server workload amount over time (c.f. Figure 9(a)) when the workload mix remains at $M2$. Figures 9(d), 9(b) and 9(c) present the variation over time of respectively the server MPL, the average client request latency and the client request abandon rate, comparing the two base systems using ad-hoc control and the CONSER-based controlled system. Notice that, due to TPC-C client think time, the number of active clients at any given time may be different from (i.e. lower than) the actual load generated by TPC-C client emulator at that time. Results show that the CONSER-based controlled MPL is able to adjust its value to the optimal value so that the performance constraint is guaranteed. Whereas in the case of the system with ad-hoc control 1, the latency grows up to 11.5 s, with an overhead of up to 44 % compared to CONSER. The system with ad-hoc control 2 allows to guarantee the performance constraint but the abandon rate grows up to 40 %, with an overhead of up to 14 %.

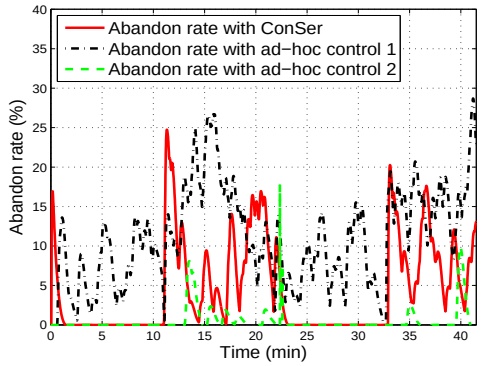
In the CONSER-based controlled system, the abandon rate is mainained at 0% with up to 70 clients. Then, the abandon rate increases with the increase of concurrent clients in the system, to attain its highest value when the number of clients is maximum, in order to keep latency below the target maximum latency. Notice that



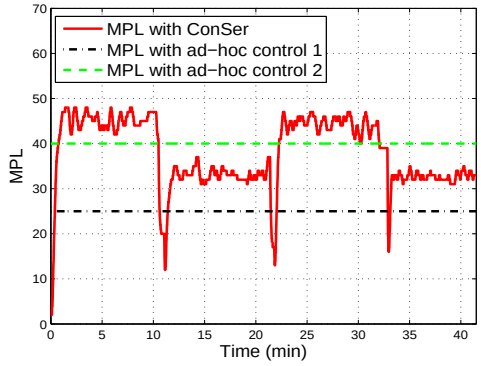
(a) Workload mix



(b) Latency

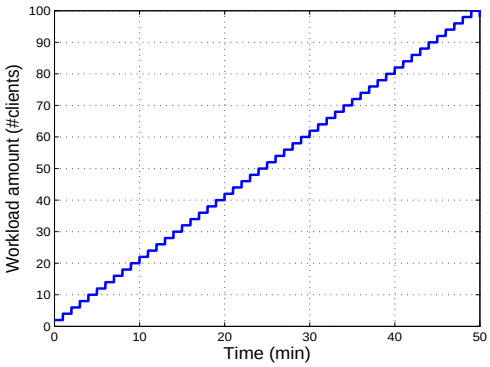


(c) Abandon rate

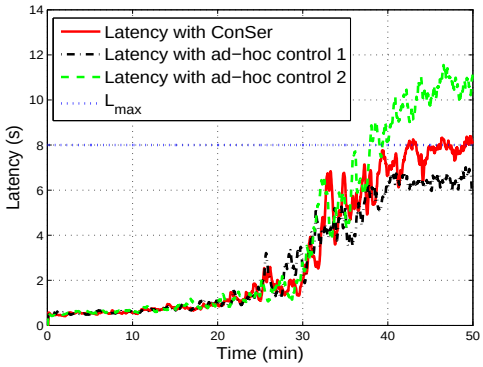


(d) MPL of controlled system

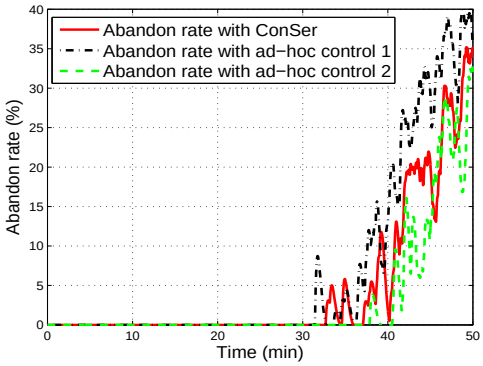
Fig. 8. System behavior upon workload mix variation – AM-$\mathscr{C}$-based controlled system vs. non-controlled system



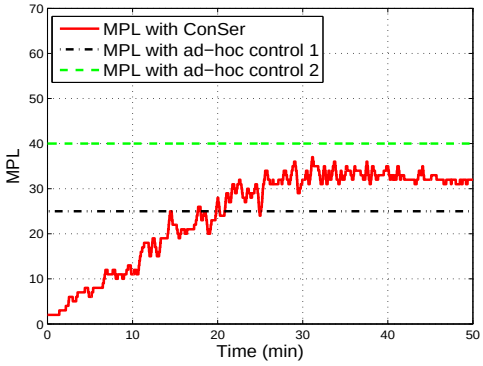
(a) Workload amount



(b) Latency



(c) Abandon rate



(d) MPL of controlled system

Fig. 9. System behavior upon workload amount variation – AM-$\mathscr{C}$-based controlled system vs. non-controlled system

at the end of the experiment (between the 40th and 50th minutes), it seems justifiable to have a high abandon rate since latency attains its maximum authorized value (c.f. Figure 9(d)) and client request rejection is necessary at that time to guarantee the latency constraint.

6.3 PM-$\mathscr{C}$ evaluation

In this section, we evaluate the proposed PM-$\mathscr{C}$ performance-maximizing feedback controller presented in Section 5.2. Here, we consider an availability constraint limiting the maximum client request abandon rate to 10%. The role of PM-$\mathscr{C}$ is thus to guarantee this availability constraint while maximizing service performance, through online feedback control of server MPL.

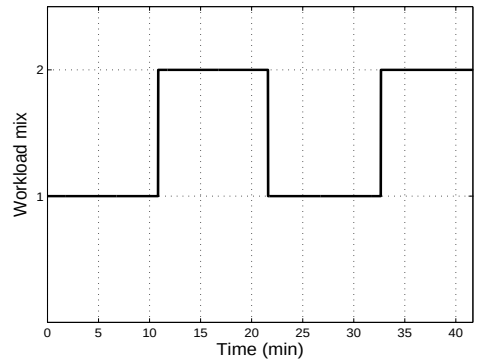
6.3.1 *Workload mix variation*

Figure 10 presents the variation of system behavior and dynamic control when the exogeneous input variable of workload mix M changes. In Figure 10(a), the workload mix varies from $M1$ to $M2$ twice when the workload amount N is of 80 clients. Figures 10(d), 10(b) and 10(c) present the variation over time of respectively the server MPL, the client request abandon rate and the average client request latency, comparing the two base systems using ad-hoc control with the ConSer-based controlled system. Here again, we notice a sudden change in the MPL when the workload mix suddenly changes, with an impact on the latency and abandon rate.

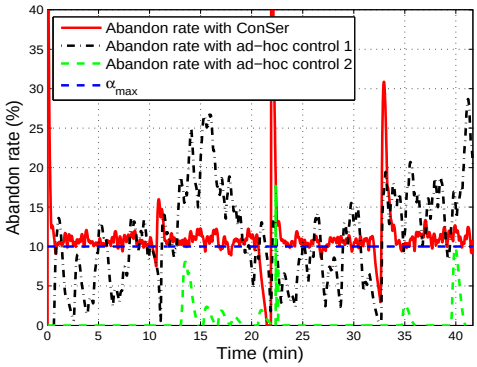
Results demonstrate that the PM-$\mathscr{C}$ controller is able to dynamically adjust MPL in order to meet the abandon rate constraint, although abandon rate is sensitive to MPL control. Under this constraint, PM-$\mathscr{C}$ keeps service performance to its maximum, with an average latency minimized to 4 s with $M1$ and to 6 s with $M2$. Whereas none of the two base systems with ad-hoc control is able to guarantee the SLOs when the workload varies. This results in an abandon rate overhead of up to 250% with ad-hoc control 1, and a latency overhead of up to 66% with ad-hoc control 2, compared to the ConSer-based controlled system.

6.3.2 *Workload amount variation*

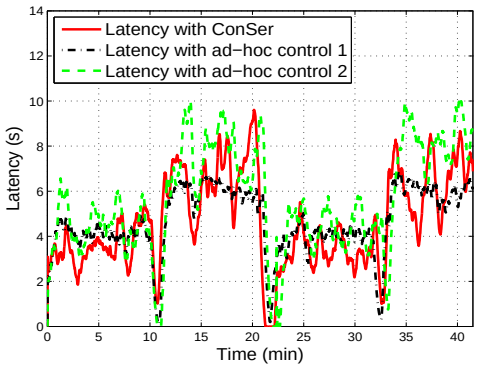
Figure 11 shows the variation of the system behavior with a varying workload amount (c.f. Figure 11(a)) and a constant workload mix $M1$. Figures 11(d), 11(c) and 11(b) present the variation over time of respectively the server MPL, the average client request latency and the client request abandon rate, comparing two base systems using ad-hoc control with the ConSer-based controlled system. Results show that the AM-$\mathscr{C}$ controller is able to adjust the MPL so that the availability constraint is met, although we can notice that abandon rate is sentitive to MPL control. During the first 15 minutes of the experiment, the ConSer-based controlled system does not reject any request since too few clients are trying to connect to the server. Compared to ConSer-based control, the base system with ad-hoc control 1



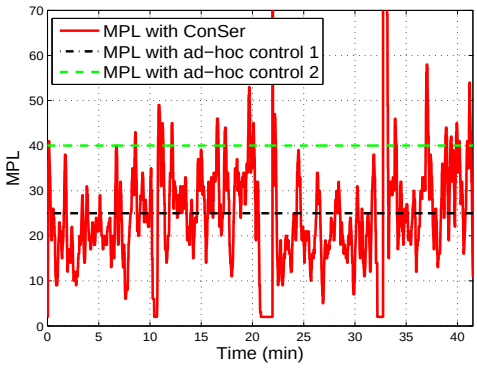
(a) Workload mix



(b) Abandon rate



(c) Latency



(d) MPL of controlled system

Fig. 10. System behavior upon workload mix variation – PM-$\mathscr{C}$-based controlled system vs. non-controlled system

(respectively ad-hoc control 2) increases abandon rate with a factor of up to 4 (respectively 3). In terms of latency, these two ad-hoc systems induce an overhead of up to 40 % (respectively 32%) compared to the ConSer-based controlled system.

7 AA-PM-$\mathscr{C}$ and PA-AM-$\mathscr{C}$ Control Laws

7.1 AA-PM-$\mathscr{C}$ availability-aware performance-maximizing control

AA-PM-$\mathscr{C}$ is another MPL control law that extends the previously presented PM-$\mathscr{C}$ law. Indeed, in section 6.3, Figure 11 illustrates the behavior of PM-$\mathscr{C}$ where the client request abandon rate is kept below a service level limit while the request latency is minimized. However, this may result in a situation where client request latency has a reasonable value (i.e. a value below a given service level limit) whereas client requests are rejected. For instance, Figures 11(b) and 11(c) respectively show that between the 14^{th} and 37^{th} minutes of the experiment, 10% of client requests are rejected while request latency is below 8 seconds. During that period of time, and if availability is prioritized over performance, availability could be maximized (i.e. rejection rate minimized) as long as performance meets a given service level objective (i.e. request latency is below a limit). Then, as long as availability objective is guaranteed (i.e. abandon rate is below a limit), performance is maximized.

Thus, AA-PM-$\mathscr{C}$ aims at guaranteeing the following tradeoff between server performance and availability, with a priority to availability as follows:

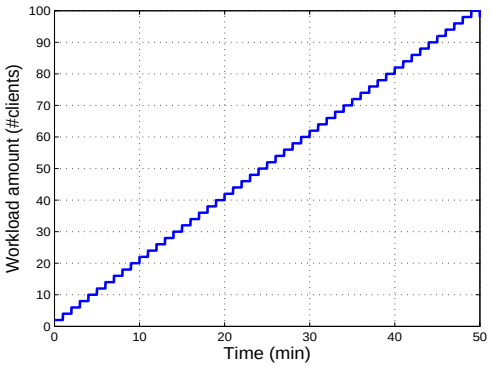
(P5) the client request abandon rate does not exceed a given maximum abandon rate $\alpha_{\max}$,
(P6) furthermore, the client request abandon rate is minimized as long as request latency does not exceed a given maximum latency $L_{\max}$, and
(P7) the request latency is minimized as long as abandon rate reaches its limit $\alpha_{\max}$.

Therefore, the AA-PM-$\mathscr{C}$ control law takes into account two limits, a request abandon rate limit and a request latency limit. This law consists in applying the AM-$\mathscr{C}$-based control when the latency is below its limit. Then if the load is too heavy to guarantee both performance and availability constraints, AA-PM-$\mathscr{C}$ switches to the PM-$\mathscr{C}$-based control.

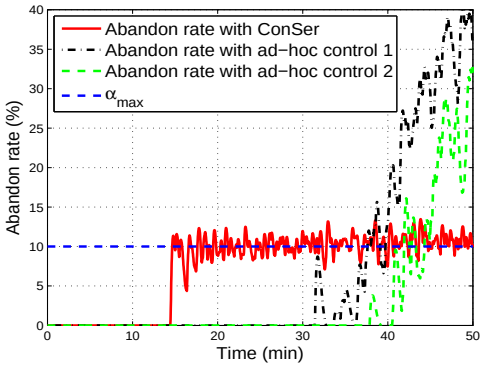
7.2 PA-AM-$\mathscr{C}$ performance-aware availability-maximizing control

Similarly, PA-AM-$\mathscr{C}$ extends the previously proposed AM-$\mathscr{C}$ law with service level limits for both performance and availability, and a priority of performance over availability. Thus, PA-AM-$\mathscr{C}$ aims at guaranteeing the following tradeoff between server performance and availability:

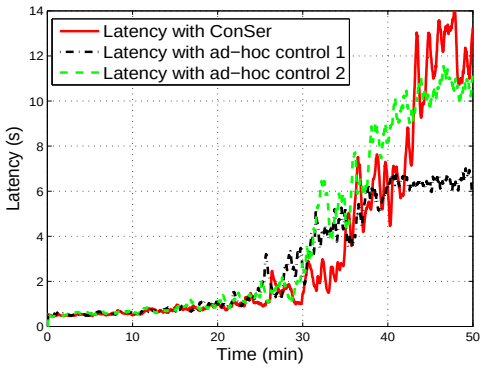
(P8) the client request latency does not exceed a given maximum latency $L_{\max}$,



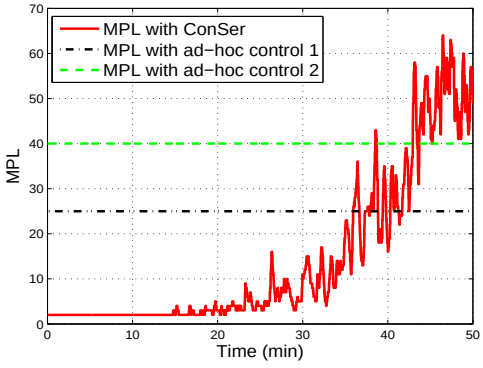
(a) Workload amount



(b) Abandon rate



(c) Latency



(d) MPL of controlled system

Fig. 11. System behavior upon workload amount variation – PM-$\mathscr{C}$-based controlled system vs. non-controlled system

(P9) moreover, the client request latency is minimized as long as request abandon rate does not exceed a given maximum abandon rate $\alpha_{\max}$, and

(P10) the request abandon rate is minimized when the latency reaches its maximum authorized limit.

Thus, PA-AM-$\mathscr{C}$ consists in first applying the PM-$\mathscr{C}$-based control when the abandon rate is below its limit. Then if the load is too heavy to guarantee both availability and performance constraints, PA-AM-$\mathscr{C}$ switches to the AM-$\mathscr{C}$-based control.

8 AA-PM-$\mathscr{C}$ AND PA-AM-$\mathscr{C}$ EVALUATION

In the following, we present the results of the evaluation of the AA-PM-$\mathscr{C}$ and PA-AM-$\mathscr{C}$ control laws and show how they improve the behavior of CONSER with respectively PM-$\mathscr{C}$ and AM-$\mathscr{C}$(c.f. Section 6). We used the same experimental environment as the one described in Section 4.1. Here again, the proposed controllers are implemented following a proxy-based approach where AA-PM-$\mathscr{C}$ and PA-AM-$\mathscr{C}$ controllers stand in front of the database server to apply online feedback control.

8.1 AA-PM-$\mathscr{C}$ evaluation

Figure 12 presents the results of the experiments conducted with AA-PM-$\mathscr{C}$ when the workload amount varies between 1 and 100 clients and the workload mix is $M2$. Here, AA-PM-$\mathscr{C}$ specifies that abandon rate should not exceed $\alpha_{\max} = 10\%$ and is reduced as long as latency remains below $L_{\max} = 8\ s$.

Figures 12(b) and 12(c) show, for instance, that during the first 35 minutes of the experiment the abandon rate with AA-PM-$\mathscr{C}$ is reduced compared to PM-$\mathscr{C}$ and does not exceed 5%, as long as latency does not exceed $L_{\max}$. When latency increases above $L_{\max}$, AA-PM-$\mathscr{C}$ provides similar behavior as PM-$\mathscr{C}$.

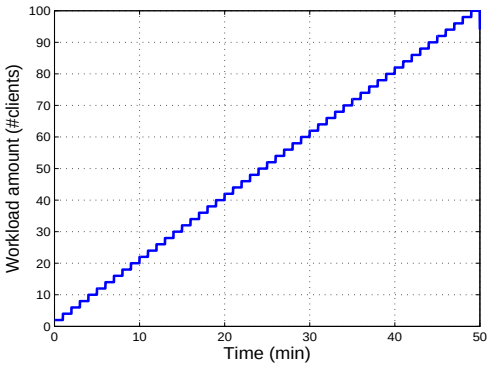
8.2 PA-AM-$\mathscr{C}$ evaluation

Experiments were conducted with PA-AM-$\mathscr{C}$ and are presented in Figure 13. They show how to improve the behavior of AM-$\mathscr{C}$. Here, PA-AM-$\mathscr{C}$ specifies that latency should not exceed $L_{\max} = 8\ s$ and is reduced as long as abandon rate remains below $\alpha_{\max} = 10\%$.

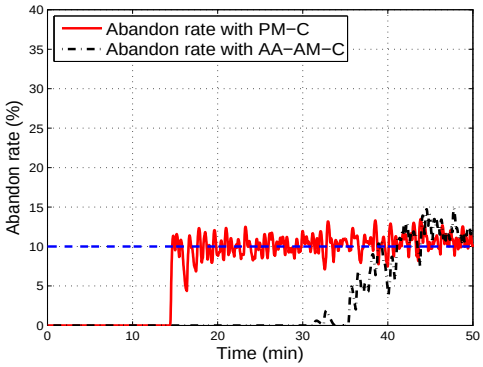
Figure 13(a) shows that the server workload amount is increasing over time while the workload mix remains at $M2$. Figures 13(b) and 13(c) show that during the first 40 minutes of the experiment the abandon rate with PA-AM-$\mathscr{C}$ remains below 10% while the latency is slightly improved compared to AM-$\mathscr{C}$. Here, latency is reduced by up to 54%.

8.3 CONSER performance overhead

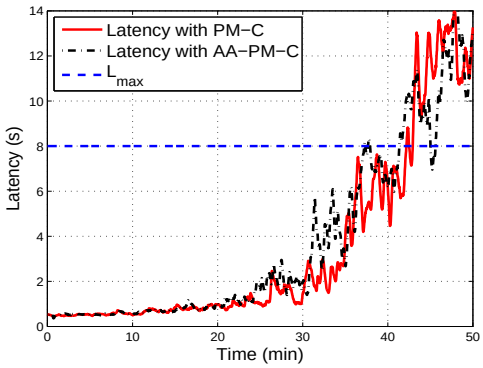
In addition to the previous evaluations, we conducted experiments to measure the performance overhead that may be induced by CONSER due to online monitoring. In the following, we compare a baseline system that does



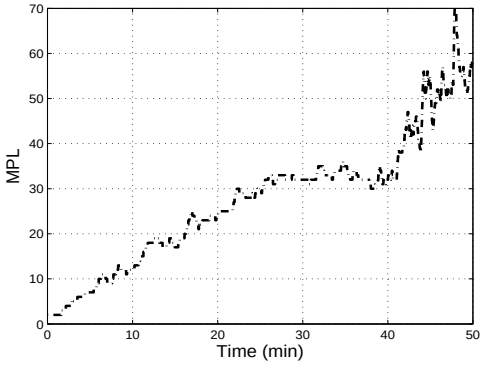
(a) Workload mix



(b) Abandon rate

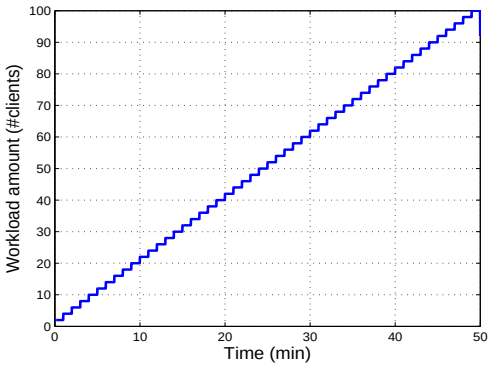


(c) Latency

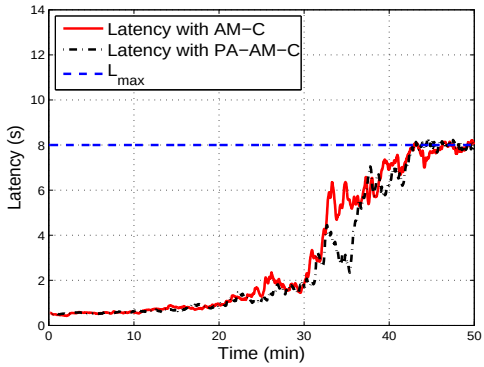


(d) MPL of controlled system with AA-PM-$\mathscr{C}$

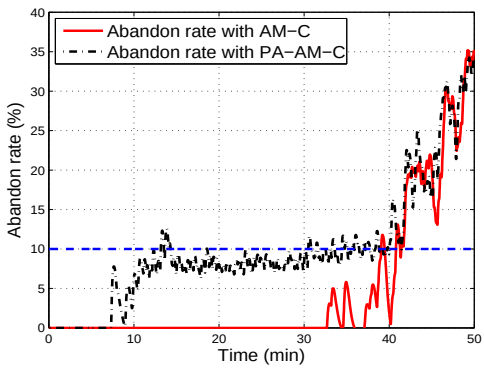
Fig. 12. System behavior upon workload amount variation – AA-PM-$\mathscr{C}$-based controlled system vs. PM-$\mathscr{C}$-based controlled system



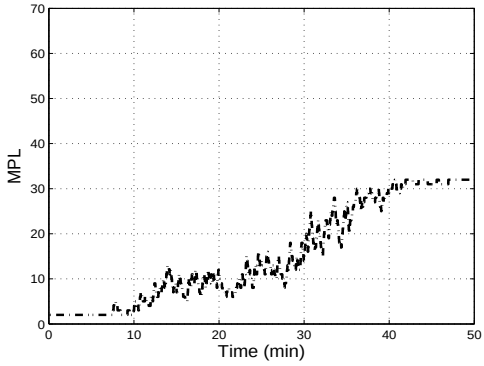
(a) Workload mix



(b) Latency



(c) Abandon rate



(d) MPL of controlled system with PA-AM-$\mathscr{C}$

Fig. 13. System behavior upon workload amount variation – PA-AM-$\mathscr{C}$-based controlled system vs. AM-$\mathscr{C}$-based controlled system

not apply control with a CONSER-based system. Both systems run the TPC-C application with workload mix $M2$ and 80 clients. Table 8.3 presents the results of this evaluation which clearly show that CONSER does not induce perceptible overhead on the application's request latency, memory usage and cpu usage.

	Baseline system	CONSER-based system
request latency	7 s	7 s
cpu usage	2%	2%
memory usage	98%	98%

TABLE 1
Performance overhead

9 RELATED WORK

This article builds upon our previous work on control of server system performance and availability, which introduced the design principles of AM-$\mathscr{C}$ and PM-$\mathscr{C}$ server control laws [26]. Here, we describe our experience with the CONSER server control prototype and, its extension with two additional control laws for multi-level optimization, at performance and availability levels, namely PA-AM-$\mathscr{C}$ and AA-PM-$\mathscr{C}$. We validate CONSER in a wide range of application workload conditions, and explore the parameter space for control accuracy and control convergence time.

Previous work has noted that system configuration is a crucial issue for the performance and availability of server systems [27], [28]. Much related work has been done in the area of system QoS management (see [19] for a good overview), investigating techniques such as session-based admission control [29], service degradation [30], service differentiation [31] and request scheduling [32]. In the following, we briefly overview the work related to admission control, and particularly MPL control, for server system management. While the improvement of server performance and availability is usually achieved by system administrators using ad-hoc tuning [11], [12], new approaches tend to appear to ease the management of such systems. Menascé et. al. propose a heuristic for the management of the QoS of servers through the determination of the multi-programming level (MPL) of servers using the hill-climbing optimization technique [13]. Although performing well in a variety of applications, hill-climbing does not guarantee optimality. In [32], a similar technique is applied; however the MPL is determined offline and thus, does not adapt to changing workloads. Other solutions to MPL identification were proposed specifically to some server technologies, such as transactional servers [33]. Other approaches aim at modeling the system in order to characterize its capacity. In [34], a simulation-based study is conducted and an analytic model is proposed to adjust server MPL according to changing workloads. However, this model is restricted to performance functions with a parabola shape and thus, does not apply to criteria such as request latency and abandon rate that usually underly service level objectives (SLOs) as perceived by clients.

Other works aiming at applying control theory to server systems appeared in the last decade. A first approach consists in applying well-known linear control theory on servers modeled as SISO (single-input single-output) or MIMO (multiple-inputs multiple-outputs) black-boxes [14], [15]. Nevertheless, due to the intrinsic non-linear behavior of these systems, linear control theory does not provide much success. Other approaches are based on non-linear models derived from queuing theory [16], [17] with a theoretical proposal in [35], [36], [18]. The resulting models interestingly predict the performance of the system, but this is obtained at the expense of a hard calibration of model parameters in order to provide accurate results. [37], [38] are other examples of the application of queuing theory models, however, they are restricted to the control of performance and do not consider availability contraints.

The proposed CONSER system differs from the previous works in many respects. It applies control theory based on fluid approximation, which results in a simpler non-linear model with very few external parameters. Fluid approximation is successfully used to model and control various systems in other areas such as car flow control and population models. In the present work, we apply it to model and control server systems, and show how this allows to provide combined guarantees on service performance and service availability.

10 CONCLUSION

This paper presents the design, implementation and evaluation of CONSER, a novel system with a nonlinear continuous-time model based on the fluid flow control theory, upon which server control is derived for server configuration. Two variants of control are primarily proposed for two different QoS objectives. AM-$\mathscr{C}$ is an availability-maximizing optimal server MPL control that achieves the highest service availability given a fixed performance constraint. PM-$\mathscr{C}$ is a performance-maximizing optimal server MPL control that meets a desired availability target with the highest performance. Two additional laws are proposed for multi-level control. AA-PM-$\mathscr{C}$ is an availability-aware performance maximizing control, and PA-AM-$\mathscr{C}$ is a performance-aware availability maximizing control. Our experiments show that the proposed techniques improve performance by up to 30 % while guaranteeing availability constraints.

While this paper concentrates on QoS metrics such as client request latency and abandon rate, we believe that both the proposed modeling and control techniques may apply to other metrics, such as server throughput. Although the proposed modeling and server control laws were applied to a database server, we believe that they could be easily applied to any sever system where MPL control holds (e.g. web servers, application servers, etc.). We also believe that the proposed control technique could be combined to other techniques such as service differentiation and degradation. Furthermore,

we are interested in how these modeling and control techniques can be applied to distributed systems.

REFERENCES

[1] Sendmail.org, 2007, http://www.sendmail.org/.
[2] Apple Inc., "QuickTime Streaming Server," 2007, http://www.apple.com/quicktime/streamingserver/.
[3] Amazon.com Inc, 2007, http://www.amazon.com/.
[4] PostgreSQL, 2008, http://www.postgresql.org/.
[5] M. Arlitt and C. L. Williamson, "Web server workload characterization: the search for invariants," *SIGMETRICS Perform. Eval. Rev.*, vol. 24, no. 1, pp. 126–137, 1996.
[6] J. A. Dilley, "Web Server Workload Characterization ," HP Laboratories, Tech. Rep. HPL-96-160, Dec. 1996.
[7] M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site," HP Laboratories Palo Alto, Tech. Rep. HPL-1999-35(R.1), Sep. 1999.
[8] Iron Mountain, "The Business Case for Disaster Recovery Planning: Calculating the Cost of Downtime," 2001, http://www.ironmountain.com/dataprotection/resources/CostOfDowntimeIrnMtn.pdf.
[9] North American Systems International Inc., "The True Cost of Downtime," 2008, http://www.nasi.com/downtime_cost.php.
[10] J. Hyman, A. A. Lazar, and G. Pacifici, "Joint Scheduling and Admission Control for ATS-based Switching Nodes," in *ACM SIGCOMM)*, Baltimore, MA, Aug. 1992.
[11] M. Brown, "Optimizing Apache Server Performance," Feb. 2008, http://www.serverwatch.com/tutorials/article.php/3436911.
[12] Microsoft, "Optimizing Database Performance," http://msdn.microsoft.com/en-us/library/aa273605(SQL.80).aspx.
[13] D. A. Menascé, D. Barbara, and R. Dodge, "Preserving QoS of E-Commerce Sites Through Self-Tuning: A Performance Model Approach," in *ACM Conference on Electronic Commerce*, Tampa, FL, Oct. 2001.
[14] S. Parekh and N. Gandhi and J. Hellerstein and D. Tilbury and T. Jayram and J. Bigus, "Using Control Theory to Achieve Service Level Objectives In Performance Management," *Real-Time Syst.*, vol. 23, no. 1-2, pp. 127–141, 2002.
[15] Y. Diao, N. Gandhi, J. Hellerstein, S. Parekh, and D. Tilbury, "Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server," *Network Operations and Management Symposium*, 2002.
[16] D. Tipper and M. Sundareshan, "Numerical methods for modeling computer networks under nonstationary conditions," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 9, pp. 1682–1695, Dec. 1990.
[17] W.-P. Wang, D. Tipper, and S. Banerjee, "A simple approximation for modeling nonstationary queues," *IEEE INFOCOM*, Mar. 1996.
[18] A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson, "Admission control for web server systems - design and experimental evaluation," *43rd IEEE Conference on Decision and Control*, Dec. 2004.
[19] J. Guitart, J. Torres, and E. Ayguadé, "A survey on performance management for internet applications," *Concurr. Comput. : Pract. Exper.*, vol. 22, no. 1, pp. 68–106, 2010.
[20] J. Lee and R. Ben-Natan, *Integrating Service Level Agreements*. Wiley, 2002.
[21] T. Abdelzaher, Y. Lu, R. Zhang, and D. Henriksson, "Practical application of control theory to Web services," *American Control Conference*, Jun. 2004.
[22] J. D. C. Little, "A proof for the queueing formula L = λ W," *Operation Research*, vol. 9, pp. 383–387, 1961.
[23] TPC-C, "Tpc transaction processing performance council," 2008, http://www.tpc.org/tpcc/.
[24] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
[25] H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.
[26] L. Malrait, S. Bouchenak, and N. Marchand, "Fluid Modeling and Control for Server System Performance and Availability," in *39th Annual IEEE International Conference on Dependable Systems and Networks (DSN 2009)*, Estoril, Lisbon, Portugal, Jun. 2009.
[27] C. Loosley, F. Douglas, and A. Mimo, *High-Performance Client/Server*. John Wiley & Sons, Nov. 1997.
[28] E. Marcus and H. Stern, *Blueprints for High Availability*. Wiley, Sep. 2003.
[29] L. Cherkasova and P. Phaal, "Session-based admission control: A mechanism for peak load management of commercial web sites," *IEEE Trans. Comput.*, vol. 51, no. 6, pp. 669–685, 2002.
[30] T. F. Abdelzaher and N. Bhatti, "Web content adaptation to improve server overload behavior," *Comput. Netw.*, vol. 31, no. 11-16, pp. 1563–1577, 1999.
[31] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, "Performance guarantees for web server end-systems: A control-theoretical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 1, pp. 80–96, 2002.
[32] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel, "A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites," in *13th international conference on World Wide Web*, New York, NY, May 2004.
[33] B. Schroeder, M. Harchol-Balter, A. Iyengar, E. Nahum, and A. Wierman, "How to determine a good multi-programming level for external scheduling," in *22nd International Conference on Data Engineering*, Atlanta, GA, Apr. 2006.
[34] H.-U. Heiss and R. Wagner, "Adaptive Load Control in Transaction Processing Systems," in *17th International Conference on Very Large Data Bases*, San Francisco, CA, 1991.
[35] M. Kihl, A. Robertsson, and B. Wittenmark, "Analysis of admission control mechanisms using non-linear control theory," *8th IEEE International Symposium on Computers and Communication*, pp. 1306–1311 vol.2, Jul. 2003.
[36] M. Kihl, "Performance Modelling and Control of Server Systems Using Non-Linear Control Theory," in *18th International Teletraffic Congress*, Berlin, Germany, Sep. 2003.
[37] A. Kamra, "Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites," in *In International Workshop on Quality of Service (IWQoS*, 2004, pp. 47–56.
[38] X. Liu, J. Heo, and L. Sha, "Adaptive control of multi-tiered web application using queueing predictor," in *in: 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006*, 2006.

Luc Malrait is a PhD candidate in Control at Grenoble Institute of Technology and conducts his research at INRIA and Gipsa-Lab. His research focuses on applying control theory to server systems. He received his MS in control engineering from Grenoble Institute of Technology in 2007.

Sara Bouchenak is Associate Professor in computer science at Grenoble University since 2004. She conducts research at INRIA and LIG Laboratory, focusing on highly-available, dependable and manageable distributed systems. Prior to that, she worked at EPFL, Switzerland, in 2003. Sara Bouchenak is an IEEE member, an ACM EuroSys member, and an officer of the French chapter of the ACM-SIGOPS. She received her PhD in computer science from the Grenoble Institute of Technology in 2001, and her MS in computer science from Grenoble University in 1998.

Nicolas Marchand is Researcher at CNRS, France. His research interests focus on the control of server systems and systems on chips with asynchronous interconnections. He received his PhD in control from the Grenoble Institute of Technology in 1999.